Assignment 3

Part 1

```
import pandas as pd
import numpy as np

```

```
data = pd.read_csv('17078.csv') # reading the given data
```

```
data
```

	f1	f2	f3	f4	f5	f6	f7	
0	-3.845862	1.724127	-3.791628	3.008085	1.054024	-0.936138	-1.515552	-3.6
1	-5.139204	-6.731594	1.064878	-4.781021	1.231421	5.254207	-3.014583	0.
2	-7.971281	-1.030925	2.904664	2.079083	6.545191	-2.841261	0.977458	-0.4
3	-2.139565	1.922493	1.775053	-4.092461	-2.011769	-1.439583	0.727577	-6.
4	-7.848439	-0.856079	2.841110	1.090115	6.680415	-2.146358	0.533774	-1.3
...	...	...	...	...	...	...	...	
2995	-5.629641	-1.612974	2.323910	3.085447	0.887611	-4.696745	2.426052	0.6
2996	-7.152639	-2.415076	4.171785	-0.148619	6.206683	0.309827	1.791623	-3.
2997	4.501343	-5.172212	-5.939482	-4.388959	1.205370	3.770760	-4.985376	0.7
2998	0.214206	-4.088398	2.641582	-1.814317	-1.062707	3.481858	-3.810297	0.
2999	1.089820	-7.938049	0.147081	-11.607528	-2.394391	7.972605	6.227041	-0.

3000 rows × 26 columns

```
data_with_label_4 = data[data.label == 4]
data_with_label_4.head()
```

	f1	f2	f3	f4	f5	f6	f7	
0	-3.845862	1.724127	-3.791628	3.008085	1.054024	-0.936138	-1.515552	-3.6528
9	1.810064	-4.010127	-9.280709	1.616508	-0.718570	1.379710	-0.369463	-2.8084
32	-4.767479	-3.626825	-3.104800	2.593647	1.174736	2.786723	1.064341	-2.0986

Selecting two classes (4 and 5)

```
data_with_label_5 = data[data.label == 5]
data_with_label_5.head(5)
```

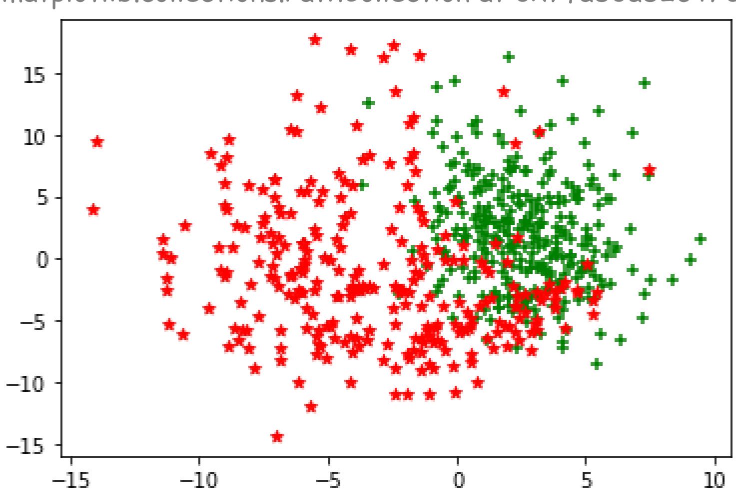
	f1	f2	f3	f4	f5	f6	f7	
21	-3.396799	-4.769202	-2.439840	2.380496	3.552324	-2.017597	1.145270	1.69(
40	2.927490	5.485425	2.794307	-5.830147	-1.908526	-5.076520	-0.921462	-2.86(
56	-4.143994	-2.673354	-0.314324	-2.301756	-1.550293	-5.241389	1.476537	0.44
101	-4.372414	-1.280444	0.675753	-1.466079	-2.761240	-5.815876	-0.702929	0.67
108	-0.659633	-7.065051	-5.149917	-8.891039	-3.594333	-2.305496	-1.159155	-4.41

Ploting scatter polt based on the f4 and f2 fro the selected class

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
plt.scatter(data_with_label_4['f4'],data_with_label_4['f2'],color='green',marker='+')
plt.scatter(data_with_label_5['f4'],data_with_label_5['f2'],color='red',marker='*')
```

<matplotlib.collections.PathCollection at 0x7fa80a82c470>




```
data_with_label_5_or_4  = data_with_label_5.append(data_with_label_4, ignore_index=True)
```

```
data_with_label_5_or_4
```

	f1	f2	f3	f4	f5	f6	f7	
0	-3.396799	-4.769202	-2.439840	2.380496	3.552324	-2.017597	1.145270	1.69
1	2.927490	5.485425	2.794307	-5.830147	-1.908526	-5.076520	-0.921462	-2.86
2	-4.143994	-2.673354	-0.314324	-2.301756	-1.550293	-5.241389	1.476537	0.44
3	-4.372414	-1.280444	0.675753	-1.466079	-2.761240	-5.815876	-0.702929	0.67
4	-0.659633	-7.065051	-5.149917	-8.891039	-3.594333	-2.305496	-1.159155	-4.41
...	...	...	...	...	...	...	...	
583	-3.402778	-0.769587	-3.093480	3.089174	-2.135419	-0.782594	0.691629	-0.55
584	1.879157	2.482861	-6.568294	3.914682	-6.011040	2.332643	2.264472	-4.21
585	-4.614316	1.371104	-2.215131	0.818999	-9.289083	-0.800645	-0.860114	0.75
586	5.583162	0.045519	-5.235267	3.677784	-4.425567	5.835448	-2.596942	-3.87
587	-3.324801	6.777687	-1.747376	1.388551	-1.047665	0.598600	4.785271	-5.87

588 rows × 26 columns

Actual training of SVM

```
from sklearn.model_selection import train_test_split
```

```
X = data_with_label_5_or_4.drop(['label'],axis='columns')
Y = data_with_label_5_or_4.label

```

```
x_train, x_test,y_train,y_test = train_test_split(X,Y,test_size=0.25,shuffle =True)

```

```
from sklearn.svm import SVC
model = SVC(C=0.5,kernel='rbf',gamma=0.01)

```

```
model.fit(x_train,y_train)
```

```
SVC(C=0.5, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
```

```
model.score(x_test,y_test)
print('Accuracy on the test set is {} %'.format(model.score(x_test,y_test)*100))
```

```
Accuracy on the test set is 98.63945578231292 %
```

Observation

As I increase the value of C the model is going to overfit to the training set and as I decrease the C value the model is going to under fit. This can be seen from the variation from the score of the model.

Increasing gamma value decreses the accuracy of model on test data and vice-versa

Talking about kernel, I chose the Radial Bias Function because it gave me the accuracy of 97% on test data. Linear,Poly,Sigmoid function also gave me nearly the same accuracy. Another reason for choosing the RBF function because in the Stanford Lecture that Our Instructor has shared with us is being talking about this function mostly.

I am able to figure out between under fitting and good fitting ,But not able to figure out between overfitting and good fitting, The reason for that is, as per the Theory as we increase the value of C the model should overfit to the training data but here as i increase the value of C to 10 or 5 the Accuracy of the model on test data tends to 1. May be this is because the test data just redundant values of the training data

Now training SVM model using only first 10 features

```
X = data_with_label_5_or_4.drop(['f11','f12','f13','f14','f15','f16','f17','f18','f19',
                                 'f20','f21','f22','f23','f24','f25','label'],axis='columns')
Y = data_with_label_5_or_4.label
```

Sperating test and train Data

```
x_train, x_test,y_train,y_test = train_test_split(X,Y,test_size=0.25,shuffle =True)
```

Defining ,training and Testing the model

```
x_train
```

	f1	f2	f3	f4	f5	f6	f7	
87	11.822444	11.358135	7.375486	-1.754605	9.410773	-0.846839	-3.228736	1.36
544	-2.490493	7.544331	-0.117851	-0.222871	-0.592519	-1.336513	3.536571	-1.7
459	20.127696	10.199229	-1.550087	6.797903	5.529289	12.088657	3.841230	7.26
397	-3.400297	3.478162	-2.065074	1.042923	-5.045301	-0.288636	0.799367	-5.4
51	-3.236968	-8.212362	-5.373748	-2.858064	2.920193	0.974947	-2.476609	-1.2
...	...	...	...	...	...	...	...	
19	1.150550	-2.843886	3.939033	2.622914	-4.076597	-5.044954	1.561273	5.4
498	-4.991092	-0.991639	-3.845838	0.725068	-5.447825	2.054813	-1.435987	-0.9
558	10.274864	11.869072	-5.254903	2.489119	2.120213	1.812496	4.740658	-4.9
403	-2.868886	4.907648	-1.185681	3.531168	2.223252	1.404702	0.911591	-1.0
117	3.939319	9.354153	3.306734	2.218801	0.424771	3.632419	4.752293	-6.9

```
model1 = SVC(C=0.5,kernel='rbf',gamma=0.01)
model1.fit(x_train,y_train)
print('Accuracy on the test set is {} %'.format(model1.score(x_test,y_test)*100))
model1.score(x_test,y_test)
```

```
Accuracy on the test set is 98.63945578231292 %
0.9863945578231292
```

The accurary of the model is same as the accuracy of the previous model. This implies that only the first 10 features of the original data set is making a difference to the training of the model, rest all coloums are no use to us.

Now selecting other two pair of classes (0 and 9)

```
data_with_label_0 = data[data.label==0]
data_with_label_9 = data[data.label==9]
data_with_label_0_or_9  = data_with_label_0.append(data_with_label_9,
                                                   ignore_index=True)
```

```
X = data_with_label_0_or_9.drop(['label'],axis='columns')
Y = data_with_label_0_or_9.label
x_train, x_test,y_train,y_test = train_test_split(X,Y,test_size=0.25,shuffle =True)

```

```
model1 = SVC(C=0.55,kernel='rbf',gamma=0.01)
```

```
model1.fit(x_train,y_train)

print('Accuracy on the test set is {} %'.format(model1.score(x_test,y_test)*100))
```

```
Accuracy on the test set is 93.10344827586206 %
```

Now training SVM model using only first 10 features

```
X = data_with_label_0_or_9.drop(['f11','f12','f13','f14','f15','f16','f17','f18','f19',
                                 'f20','f21','f22','f23','f24','f25','label'],axis='columns')
Y = data_with_label_0_or_9.label
x_train, x_test,y_train,y_test = train_test_split(X,Y,test_size=0.25,shuffle =True)
```

```
model1 = SVC(C=0.55,kernel='rbf',gamma=0.01)
model1.fit(x_train,y_train)
print('Accuracy on the test set is {} %'.format(model1.score(x_test,y_test)*100))
```

```
Accuracy on the test set is 99.3103448275862 %
```

The acuuracy of the model has increased as we have decreased the number of features coulumns

Now selecting other two pair of classes (2 and 7)

```
data_with_label_2 = data[data.label==2]
data_with_label_7 = data[data.label==7]
data_with_label_2_or_7  = data_with_label_2.append(data_with_label_7,
                                            ignore_index=True)
```

```
X = data_with_label_2_or_7.drop(['label'],axis='columns')
Y = data_with_label_2_or_7.label
x_train, x_test,y_train,y_test = train_test_split(X,Y,test_size=0.25,shuffle =True)
```

```
model1 = SVC(C=0.55,kernel='rbf',gamma=0.01)
model1.fit(x_train,y_train)
print('Accuracy on the test set is {} %'.format(model1.score(x_test,y_test)*100))
```

```
Accuracy on the test set is 93.63057324840764 %
```

Now training SVM model using only first 10 features

```
X = data_with_label_2_or_7.drop(['f11','f12','f13','f14','f15','f16','f17','f18',
```

```
                    'f19','f20','f21','f22','f23','f24','f25','label'],axis='columns')
Y = data_with_label_2_or_7.label
x_train, x_test,y_train,y_test = train_test_split(X,Y,test_size=0.25,shuffle =True)
```

```
model1 = SVC(C=0.55,kernel='rbf',gamma=0.01)
model1.fit(x_train,y_train)
print('Accuracy on the test set is {} %'.format(model1.score(x_test,y_test)*100))
```

```
Accuracy on the test set is 98.72611464968153 %
```

The accuracy of the model has increased as we have decreased the number of features coulumns

Multiclass Classification

```
data.head(10)
```

	f1	f2	f3	f4	f5	f6	f7	
0	-3.845862	1.724127	-3.791628	3.008085	1.054024	-0.936138	-1.515552	-3.6528
1	-5.139204	-6.731594	1.064878	-4.781021	1.231421	5.254207	-3.014583	0.3151
2	-7.971281	-1.030925	2.904664	2.079083	6.545191	-2.841261	0.977458	-0.4653
3	-2.139565	1.922493	1.775053	-4.092461	-2.011769	-1.439583	0.727577	-6.2550
4	-7.848439	-0.856079	2.841110	1.090115	6.680415	-2.146358	0.533774	-1.3485
5	-5.138542	5.377690	-1.554974	1.468151	-1.779365	-0.689109	2.202427	-5.4591
6	1.711062	-1.803268	10.568383	4.380619	-1.951835	6.095458	1.521579	-2.4582
7	-3.005429	0.046924	2.761363	4.566784	-0.107491	1.813141	1.841967	-2.2794
8	12.092873	-10.138759	-4.918328	-3.628506	-1.401219	-0.915427	3.840340	2.6722
9	1.810064	-4.010127	-9.280709	1.616508	-0.718570	1.379710	-0.369463	-2.8084

```
data = pd.read_csv('17078.csv')
X = data.drop(['label'],axis='columns')
Y = data.label
x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size=0.25,shuffle = True)
```

```
model2 = SVC(C=1, kernel='rbf',gamma = 0.01 )
model2.fit(x_train,y_train)
print('Accuracy on the test set is {} %'.format(model2.score(x_test,y_test)*100))
```

Accuracy on the test set is 90.26666666666667 %

When we have a model like

```
SVC(C=1, kernel='rbf',gamma = 0.00001 )
```

The Accuracy is 30% This means the model underfit.

When the model is like

```
SVC(C=1, kernel='rbf',gamma = 0.1 )
```

The accuracy is 41.4% , This is a case of overfitting

But When the model is

```
SVC(C=1, kernel='rbf',gamma = 0.01 )
```

I get an accuracy of 90.4% This means its a good fitting case.

All this observation agrees with the Theory. As we increase the gamma the the model starts to overfit and vice-versa.

Now training SVM model using only first 10 features

```
X = data.drop(['f11','f12','f13','f14','f15','f16','f17','f18','f19','f20','f21','f22',
              'f23','f24','f25','label'],axis='columns')
Y = data.label
x_train, x_test,y_train,y_test = train_test_split(X,Y,test_size=0.25,shuffle =True)
```

```
model2 = SVC(C=1, kernel='rbf',gamma = 0.01 )
model2.fit(x_train,y_train)
print('Accuracy on the test set is {} %'.format(model2.score(x_test,y_test)*100))
```

```
Accuracy on the test set is 88.66666666666667 %
```

The multiclass support is handled according to a* **one-vs-one scheme.***

The Accuracy decreases as we decreased the number of feature coulunms, this tells that all features is important for training the SVM

Part 2

```
f.close()
```

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.

```
data2 = pd.read_csv('train_set.csv',header=None)

```

```
X = data2.iloc[:,0:25]
Y = data2.iloc[:,25]
data2.head()
```

	0	1	2	3	4	5	
0	422.873896	171.642843	777.802417	791.759675	-230.159885	620.868152	257.4
1	247.380707	-512.142886	-1245.178446	-258.132575	72.679989	206.472667	17.2
2	596.307191	1286.992777	-100.349549	-412.271616	-496.773516	-60.343716	243.9
3	-467.027604	-37.242249	-497.042279	-530.977693	-33.318369	306.761390	-58.7
4	622.065913	299.733778	679.166007	677.085278	427.154720	517.270666	394.7

```
x_train, x_test,y_train,y_test = train_test_split(X,Y,test_size=0.25,shuffle =True)

```

```

model3 =SVC(C=8)
model3.fit(x_train,y_train)
model3.score(x_test,y_test)
```

```
0.9704
```

```
test_data = pd.read_csv('test_set.csv',header=None)
arr = model3.predict(test_data)

```

```
print(arr)
```

```
[3 5 2 ... 7 2 5]
```

```
import csv
fields = ['id','class']
filename = 'result.csv'
f = open (filename,"w",newline="")
writer = csv.writer(f)
writer.writerow(fields)
for i,item  in enumerate(arr):
  tup=(i,item)
  writer.writerow(tup)

```